Adapting Agile Methodology to Overcome Social Differences in Project Members

Hitoshi Ozawa and Lan Zhang
OGIS RI Co., Ltd.
Tokyo, Japan
Ozawa_Hitoshi@ogis-ri.co.jp, Zhang_Lan@ogis-ri.co.jp

Abstract— Projects often consist with members with different values that may cause conflicts within the team causing decrease in members' motivation, involvement, and cohesiveness. In our experiences with offshoring Japanese software development projects to China, we were having difficulties with low quality deliverables and high turnover rate of Chinese members because of social differences. Our attempts to create a common culture were not very successful because people in general are less likely to change their basic views and behavior in a short period of time. We, however, were able to obtain success by acknowledging that differences are going to exist and adopting and adapting agile practices in consideration of the existence of these differences.

We will show Kaizen as is used by a Japanese company in software development. We will focus on our experiences with social differences we've found and how we continuously adapted practices in our project to take better advantage of the situation as relationship between members changed. It is based on our over 10 years of experience in trying to improve a software package development at a software company in China, which has now become our subsidiary. During our attempts, we have learned the importance of agile mentality in resolving value difference issues. We believe what we've learned in adapting agile practices is not just limited to our particular project but can be useful in agile projects in general and thus can be used to assist resolve value differences in organizations as well.

Keywords— Scrum; outsourcing; offshore; distributed project; socioculture; global software development; communication; collaboration

I. Introduction and Background

OGIS-RI is a wholly owned subsidiary of Osaka Gas specializing in offering IT services to the parent company and to other large Japanese corporations. OGIS-RI has begun experimenting with offshoring projects in China from market demand. Executives were behind the project and we had full support to make outsourcing succeed.

This paper is based on our findings and lessons learned based on our 10 years of experience in adapting our offshore projects to a Chinese software company (currently our subsidiary - Shanghai OGIS Tonghua Software Co., Ltd.).

II. Sociocultural Differences

Software development is human centric and socio-technical activity. Complex interaction of values, attitudes, behavioral norms, beliefs, communication approaches by members of a project with vastly different values may give rise to misunderstanding and misinterpretation of intent that may result in conflict, mistrust, and underutilization of talents. Collocated projects may have members with different values but it is more commonly an issue in offshore projects because value differences of social differences among members are much more apparent.

Japan and China societies seem to share many common values such as hierarchical social structure and emphasis on social network but have subtle differences. These differences may be the result of Japan being an island with agricultural society with most citizens not accustomed to other culture while China is part of a continent and with society more accustomed to diversified cultural exchange.

A. Hofstede's Cultural Dimensions

Geert Hofstede identifies social culture dimensions that can be used to measure differences in how society prioritizes some basic social concepts. Chinese and Japanese index values are close in concepts of individualism - collection, power distance, and long-term orientation [5]. However, China and Japan differ in the uncertainty avoidance index (UAI) - the level of tolerance for uncertainty and ambiguity within the society. As shown in Table 1, Japan has a very high UAI value of 92 while China has a very low value of 30.

TABLE 1. Hofstede's cultural dimensions indices of Japan and China

	Japan	**China**
Individualism-collectivism	46	20
Power distance	54	80
Uncertainty avoidance	92	30
Masculinity - Femininity	95	66
Long-term orientation	80	118

A culture like Japan with high uncertainty avoidance presents little tolerance for ambiguity and prefers detailed planning compared to a culture like China with lower uncertainty avoidance index.

High uncertainty avoidance index does not imply people are less interested in new technology but are less prone to adopt it within an actual project. For example, many Japanese engineers are interested in new technology and sales of books on new technology are very high. For example, books concerning Scrum ranked 2nd and 3rd in Shoeisha, a major Japanese IT publisher, computer book ranking (February 2013). Nevertheless, the adoption rate of agile methodology is only 2.4% [7] compared to 35% in the US [4].

In offshoring software project, this subtle difference in view of risk cause conflicts between Japanese members and Chinese members. Japanese members emphasized more on reducing "risk" and on improving software quality using "proven" practices. They were also more tolerant of conflicts because they are less likely to change jobs. Chinese developers, however, were more interested in using new technology, and they were also more likely to take a "risk" by changing jobs when a conflict occurred.

B. Sociocultural Challenges

During our offshoring experience with a company in China, we detected three Sociocultural challenges related to risk avoidance: (1) openness of society, (2) difference in willingness to adopt new techniques and technology, and (3) problem encountered with implicit communication over explicit communication.

1) Difference in Openness of society

As is typical of a society with high value of UAI, Japanese society is more closed and close knit. People do not move much and still hold community gatherings as well as neighborhood activities such as daily trash cleanup and neighborhood circular notices. In a similar manner, Japanese employees think of employment as a lifetime commitment to a company. Japanese companies prefer to recruit fresh graduates from school and “teach” them to fit with their group instead of hiring experienced people who likely have their own ways of doing things. Gaining consensus to reduce conflict is valued than gaining higher skills to obtain better position. On the other hand, Chinese members tended to switch job after acquiring necessary skills for a new position. This created confusion and conflict between Japanese and Chinese members.

2) Difference in willingness to adopt new techniques and technology

Japanese society and organization places more emphasis on avoiding failure instead of attaining success. Therefore, they are more likely to use existing methodology and technology and only adopt minor changes. On the other hand, Chinese members were more interested in acquiring new skills by trying new techniques and technology. Difference in organization cultural rate in adopting new techniques and technology caused friction between the organizations.

3) Differences in Communication

There is a difference in implicit knowledge implied by Japanese and Chinese cultures. Miscommunication and misunderstanding are caused mainly by differences in interpretation caused by differences in social context rather than from syntactical differences. Implicit knowledge is socially embedded within social norms and not easily shared out of context. Japanese tend to have more "implicit" unwritten meanings in sentences because Japanese members usually form a very tightly closed group where members understand each other without explicitly saying everything.

III. *Methodology*

Issues in a project can be classified into 3 types as shown in Table 2: (1) anticipated issue with knowable outcome, (2) anticipated issue with an unknown outcome, (3) unanticipated issue. Furthermore, the outcome can further be divided into (a) outcome that can be resolved, (b) outcome that cannot be easily resolved within the current situation, and (c) problems that do not affect project outcome.

TABLE 2. ISSUE ANALYSIS TABLE

	(1)Difference anticipated with known outcome	**(2)Difference anticipated with unknown outcome**	**(3)Difference unanticipated**
(a)Can change			
(b)Cannot change			
(c)Does not affect project outcome			

As an example, there are some characters such as 以上 which are common to both Chinese and Japanese but have different meanings. To the Chinese, it means greater than while to the Japanese, it means equal to or greater than. Characters such as these are known in advance and are known to affect developed software. However, they can be circumvented during the translation process.

On the other hand, it is known there are some specifications that are implicitly defined and differ between the two cultures, but cannot be determined beforehand. In our case, error handling was expected to be implemented by the Japanese members and was not explicitly written in the documents. Chinese member, however, only implemented what was written in the specification and did not implement error handling.

An example of unanticipated difference was the high turnover from lack of motivation of the Chinese members. Lifetime employment is customary in Japanese society and turnover by project members were not expected.

Hofstede's cultural dimensions are a good model to start when analyzing culture but in an actual project, member interaction does influence each other and the cultural dimensional values change over time. To take advantage of these changes, situations were re-analyzed and roles and processes were adapted to resolvable problems after each cycle. Factors influencing problems classified as that cannot be changed where investigated to see if they still cannot be changed or if they can now be changed if practices and conditions were changed.

IV. Lessons Learned During Waterfall Methodology

Initial offshore development was done using a waterfall methodology because the Japanese members had experience with it and felt more comfortable. Most common anticipated problems encountered in offshore development such as problems arising from geographical dispersion, different time zones, and common problems associated with difference in languages were resolved during this stage.

The cost to develop a software was reduced as expected but the quality of the software was found to be unacceptable during the acceptance test. Chinese members were also reluctant to fix the problem because the Chinese society associates “concession” with willingness to compensate. Japanese member just wanted it fixed but Chinese members were reluctant to fix it because they thought it also required some compensation. Resubmitting a fixed specification and redeveloping the software until quality were satisfactory required time and expense such that it resulted in no cost benefit of offshoring a project.

Upon analysis, following 4 reasons were cited as the cause of the problem: (1) specification was ambiguous, (2) there was a cultural difference in quality acceptance level, (3) information sharing was insufficient, and (4) low motivation level in Chinese members.

Japanese members usually work in a tightly closed group and understand each other's requirements without having to express them explicitly. It is only necessary to explicitly write differences from their implicit implied norms. However, it is necessary to write a detailed specification to get the software they wanted when asking Chinese members to develop a software system. This added time and cost of creating documents that were unnecessary when developing in Japan.

Japanese customers request very high quality from the beginning. Chinese developers and testers, on the other hand, were conducting tests on normal conditions only without any test on error nor abnormal conditions. They only conducted them when the Product Owner found the software was not behaving as expected even after it passed testing.

In Japan, when the specification is ambiguous, it is customary to ask the person who wrote the specification to clarify the meaning. Chinese members view questioning as a humiliation. Thus, even when the specifications are ambiguous, they would try to "resolve" the problem based on their cultural norms, knowledge, and experience rather than contacting the person who wrote the specification.

To alleviate this problem, Japanese members were asked to write detailed complete specification explicitly. The specification was reviewed to weed out miscommunication, misunderstanding, and any ambiguities. This required considerable time and cost and Japanese members became wary of writing further detailed specifications. They complained about having to write detailed specifications but diligently completed because Japanese usually do not sway out from the current group they are in. On the other hand, the turnover rate of Chinese developers increased from lower motivation.

V. Adopting Scrum

We were able to learn and fix several sociocultural challenges during waterfall software development, but the quality of the finished software was still up to the expected level and our customers were not really satisfied. The main cause of the problem was detected as being members of the development team not fully understanding the specification. Both Chinese and Japanese educational systems have a teacher talking and students just listening. It is seen as a "humiliation" to say they did not fully understand what the teacher is talking about. This common characteristic, however, was having a negative effect because Chinese Development Team members were reporting they were not having any problem to avoid “humiliation” of saying they didn’t fully understand the specification.

Both Japanese and Chinese members were unsatisfactory with how the project was progressing and Chinese employee turnover rate was high. Motivation of Chinese members deteriorated very quickly when a key member left the company.

Both Japanese side and Chinese side agreed to use Scrum in a hope a new methodology and practice will improve the situation but with different reasons. Japanese Product Owner is held accountable for the finished deliverables and wanted to reduce risk by having shorter development cycles and more transparent status reports. Chinese members, on the other hand, wanted more authority on development so they would be more able to adopt new technologies. After Scrum was adopted, Chinese Team decided to use XP practices during each sprint.

A. Roles

All our customers were Japanese, so a role of product owner was assigned to a Japanese member residing in Japan. Roles of scrum master and the development team were assigned to the Chinese members in China. The Chinese Scrum Master was chosen to because a Scrum Master needs to communicate frequently with the Development Team to find and remove any obstacles a development team member may have.

To reduce misunderstanding and miscommunication between Japanese Product owner and Chinese scrum master, a role of Product Owner Proxy was created to bridge implicit communication gap between Japanese Product owner and Scrum Master and development team. Similarly, the role of a liaison between Japanese Product Owner and Chinese Scrum Master was assigned to a Chinese member with deep understanding of both cultures and languages. It is the responsibility of a Product Owner Proxy to act as a liaison between the Product Owner on-site with the Development Team and to make sure there were no contextual misunderstandings about the project expectation between Japanese Product Owner and Chinese Scrum Team and to arbitrate conflicts when they do.

Ideally, it would be good to have Product Owner be collocated with scrum master and development team and be part of the sprint, but we were not able to do that because the Japanese Product Owner did not have sufficient understanding of Chinese culture and command of the language and the expense to send a member to a Chinese office for an extended period of time was too high.

B. Process

Agile development process initially followed those used during waterfall development. Japanese product owner gathered requirements from customers in Japan with which user stories were written. User stories were sent to the Chinese Product Owner Proxy who reviewed the user stories and rephrased sentences that may cause misunderstanding or miscommunication.

Fig. 1 shows the communication between roles. A Japanese Product Owner who wrote the user stories went to the Chinese office for 3 days to explain the specification to the Chinese Product Owner Proxy, Scrum Master, and to the Development Teams. Chinese Development Team wrote backlogs from these user stories that were again reviewed by the Product Owner Proxy to make sure there were no misunderstandings or miscommunication.

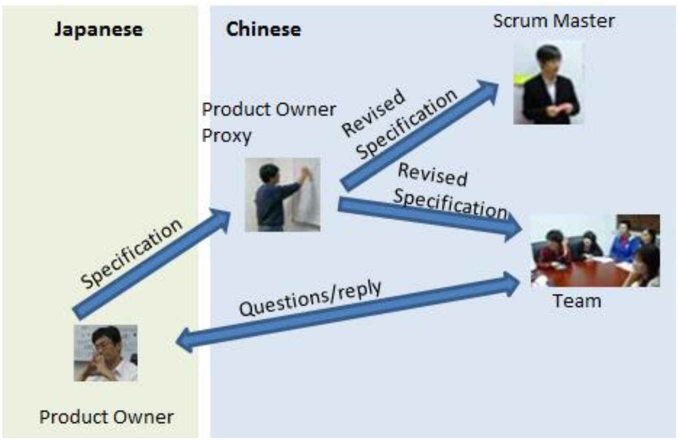

Fig. 1. Communication between roles

Product Owner, Product Owner Proxy, and the Team worked to convert user stories to the Product Backlog and then to Sprint Backlog. Software was developed by the development team based on this Sprint Backlog with a sprint cycle of 2 weeks.

Product Owner is responsible for clarifying the user stories during the planning and during the development stages. If the Development Team needs any clarification of the Product Backlog during development, they contacted the Product Owner. Questions were initially thought to be answered by the Product Owner Proxy collocated with the Development Team, but it was decided to send to the Production Owner directly instead of to the Product Owner Proxy to alleviate responsibility issues when a defect was detected.

Developed software was tested in by the Chinese testing team and when it passed all tests, the software was delivered to Japanese Product Owner who conducted acceptance tests. A Japanese Product Owner sometimes went to China to join the testing team to see the quality before conducting an acceptance test on his own.

C. Adapting Scrum

After each development cycle, agile practices were re-evaluated during agile retrospective. KPT (Keep, Problem, Try) sessions were held. Instead of just looking back at events, the current situation between project members were analyzed and detected and predictable problems were written into the "Issue analysis table" (Table 2). Those that can be changed were planned for implementation in the next cycle. Those that cannot be were further analyzed to find factors and possible solutions were proposed. If all members agree to the solution, it was implemented. If there was a disagreement, another solution was thought or it was marked as irresolvable at a current time.

The problem of development team members fully not understanding the specification persisted in our project. When we were using the waterfall methodology and during our initial attempt at Scrum, detailed specification was written so developers will be able to code programs without fully understanding the specification. Writing detailed specification required enormous time but let to an activity based mentality rather than value based mentality and resulted in Chinese member just mechanically rewriting step by step specification to computer code. Nevertheless, the quality of the deliverables was still not what we expected.

Instead of trying to make the document clearer, we rethought about the problem and concluded that the problem was not that the documents were unclear; it was that the members of the project were not fully communicating with each other and trying to understand the specification thoroughly.

Our initial attempt was to provide the documents beforehand to the development team and to request members of development team explain back in their own words to the product owner. If there was any misunderstanding, it was corrected before Sprint Backlog was written. This only proved successful to a limited extent because the time to review understanding by development members was limited so not all details were verified. Some detailed feature specifications were found to still be ambiguous during development and Product Owner were still not being contacted to resolve these issues.

Counter to intuition, our next decision was to make the initial user stories very vague so that the Development Team would be induced to contact the product owner to find the actual detailed meaning. Scrum recommends Product Owner to be responsible for expressing user stories clearly, but taking this to mean to clearly express all user stories clearly in a limited period of time in the early stage of sprint planning was actually hindering the Development Team from fully understanding the Product Backlog to the level needed because it was leading to the lessened requirement to communicate. Our initial thought, also, was that specification needed to be fully written to prevent misunderstanding because of differences in languages and members were not really interested in learning a new language.

By making the user stories very vague, Team members were forced to ask the Product Owner for clarification. This initiated much closer interaction between Japanese and Chinese members and broke the ice between the two cultural groups. Communication became much more frequent and defects were detected and fixed earlier. Members were also trying to learn each other's language and increased communication was assisting them in this effort.

Vague specification, however, requires the Product Owner to spend more time to reply to questions from the development team. This increase can be lessened if a Product Owner actually creates a regular detailed specification along with the vague user stories. Detailed specification may be consulted to reply to developers' questions or updated when a better alternative solution is found.

The problem now was adjusting the granularity of the initial specification so the questions would not overwhelm the product owner. Number of questions to verify specification were initially very high because documents contained many implicit understandings. Number of question decreased as we adjusted granularity of specification. Granularity of documents should be adjusted to initially promote communication and later to reduce cost.

Japanese members initially consented adopting Scrum to hold more periodic reviews and to better track development progress, but later shifted to give the Chinese group more autonomy because they seem to know more about what motivated them and had a better understanding between members. We, also, found Chinese members were actually able to contribute much more to making the finished software much better because of the accumulated knowledge.

VI. Results

Employee retention has been a major problem in offshore development. Employees tended to switch job after gaining skills to find another job. Time to recruit and train new employees, increase skill level, and to have a new member settle in with the team were a main problem because it took time, incurred cost, and decreased qualities of the deliverables.

After adopting and adapting agile methodology in projects, interaction between members increased. This resulted in better understanding between team members and increased members' motivation. Employee turnover rate from member dissatisfaction decreased from 20% to 0% and quality of the software became more stable across projects by retention of skilled members.

Increased communication led to increased trust between members. As members began to know each other more, it was found that development team members actually knew much more than were originally expected. Giving more responsibilities to such members resulted in a much better software.

We are still not completely satisfied with the current outcome. Members are still not completely proficient in each other's language but members now see this as their own issue that needs to be resolved. On the other hand, we still have not been able to resolve the issue surrounding QCD (quality, cost, delivery). Japanese members still are requesting a detailed report while Chinese members complained about requiring too much resource to write the reports. No suitable solution has yet been found but as the process and practice stabilize, we are trying to find a negotiation point where all members can agree on.

VII. Lessons Learned

Just adopting agile practices was not sufficient to succeed in an offshoring project. Our experience taught us the following lessons.

Lesson 1: Continuously adapt roles and processes based on the current conditions. If the relationship between members change, modify process and tasks assigned to roles to take advantage of it. For example, review members and organizations trust after a sprint or during the sprint retrospective and modify roles and process accordingly.

Lesson 2: Analyze and find factors concerning differences that cannot be changed rather than trying to find the reason for the differences. Asking members directly for a reason to a problem may not lead to a possible resolution. Chand provides an example where workers of an Indian team did not cite culture differences as the problem while managers did [1].

Lesson 3: Don't try to force to solve all difference issues. Assess how much the difference affects the result of the project. Some differences cannot be overcome because they are inherited deeply into the culture. Accept those differences and try to find a way to take advantage of it instead of trying to eliminate it.

Lesson 4: Try to resolve issues few at a time instead of trying to solve them all at once. Give time to allow members to build better relationships based on the change.

Lesson 5: Don't give up. Even if difference issues cannot be resolved now, it may be resolved as project environment change. Always keep changing the project environment so issues may become resolvable in the future.

VIII. Conclusion

We were able to go beyond the difficulties of offshore development and were able to gain advantages by adopting and adapting agile practices. We were able to overcome social differences in project members by promoting communications between members. We adopted Scrum, but it wasn't just assigning roles and process that really improved the project - it was creating an environment and practices where members were urged to communicate more and understand each other.

Some of the practices we found successful may not work in other environments, but our experience of continuously adapting roles and practices to take advantages of changing the relationship between members seems to be a general rule that can be followed. It was not just enough to define a process and setup tools to encourage better communication between project members - it was necessary to change our practices as well to encourage members to take advantage of them. As a result, we were able to increase the quality of the created deliverables with members more satisfied with the project.

References

[1] D. Chand, G. David, R. Galliers, S. Kumar, "An Investigation of How Culture Impacts Global Work: Unpacking the Layers of Culture".

[2] M. Cohn, “Succeeding with Agile: Software Development Using Scrum”, Addison-Wesley Professional, 2009.

[3] S. Deshpande, Valentine Casey, Ita Richardson, Sarah Beecham, "Culture in Global Software Development - a Weakness or Strength?".

[4] Forrester Research Inc. 2010 "Agile Development: Mainstream Adoption Has Changed Agility".

[5] G. Hofstede, "The Hofstede Centre", http://geert-hofstede.com/

[6] Y Hsieh, P. Kruchten, E. MacGregor, "Matching Expectations: When Culture Wreaks Havoc with Global Software Development".

[7] IPA/SEC White Paper 2012 on Software Development in Japan (Information -technology Promotion Agency, Japan) p.42 figure 4-5-1.

[8] M. Poppendieck, T. Poppendieck, "Leading Lean Software Development", Addison-Wesley Professional, 2010.